

```
#include <stdio.h>
```

```
// Function to heapify a subtree rooted at index i
```

```
void heapify(int arr[], int n, int i) {  
    int largest = i;          // Initialize largest as root  
    int left = 2 * i + 1;     // Left child index  
    int right = 2 * i + 2;    // Right child index  
  
    // If left child is larger than root  
    if (left < n && arr[left] > arr[largest])  
        largest = left;  
  
    // If right child is larger than largest so far  
    if (right < n && arr[right] > arr[largest])  
        largest = right;  
  
    // If largest is not root  
    if (largest != i) {  
        int temp = arr[i];  
        arr[i] = arr[largest];  
        arr[largest] = temp;  
  
        // Recursively heapify the affected subtree  
        heapify(arr, n, largest);  
    }  
}
```

```
// Main function to perform heap sort
```

```

void heapSort(int arr[], int n) {
    // Build a max heap
    for (int i = n / 2 - 1; i >= 0; i--)
        heapify(arr, n, i);

    // One by one extract elements from heap
    for (int i = n - 1; i > 0; i--) {
        // Move current root to end
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;

        // Call heapify on the reduced heap
        heapify(arr, i, 0);
    }
}

// Function to print the array
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Driver code
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};

```

```
int n = sizeof(arr) / sizeof(arr[0]);
```

```
printf("Original array:\n");
```

```
printArray(arr, n);
```

```
heapSort(arr, n);
```

```
printf("\nSorted array (Heap Sort):\n");
```

```
printArray(arr, n);
```

```
return 0;
```

```
}
```

Original array:

12 11 13 5 6 7

Sorted array (Heap Sort):

5 6 7 11 12 13

=== Code Execution Successful ===